# GIPAD – GS2
# Recherche Opérationnelle - APP2
# Correction devoir final

21 novembre 2011

---

**durée :** 2h30

**documents autorisés :** tous documents papiers, hors livres et photocopies de livre.

**barème :** note sur 20 = nombre de points * 20 / 30

1. Programmation dynamique (10 points)
2. Modélisation PLNE et Heuristiques (20 points)

---

## 1 Programmation dynamique

**Problème 1** Partition.

```
boolean count(int[] v, int m, int p) {
   boolean[] c = new boolean[m];
   for (int k=0; k < c.length; k++) {
     c[k] = false;
   }
   c[0] = true;
   for (int i=0; i < v.length; i++) {
     for (int k=0; k < c.length; k++) {
       if (c[k] && (k+v[i] < m)) {
         c[k+v[i]] = true;
       }
     }
   }
   return c[p];
}
```

**Question 1** algorithme ***(5 points)***

**Q1.1.** déterminer la complexité et la fonction de l'algorithme `count` en java ci-dessus.

**Question 2** partition ***(5 points)***
*Étant donnés un ensemble* $S$ *d'objets, chaque objet* $i \in S$ *ayant un poids* $w_i \in \mathbb{N}_*$*, le problème de partition est le problème de décider s'il existe une partition de* $S$ *en deux sous-ensembles de poids identiques.*

**Q2.1.** montrer comment le problème de partition peut être résolu au moyen de l'algorithme `count` ;

**Q2.2.** en déduire la classe de complexité de ce problème ;

**Q2.3.** modifier l'algorithme de manière à retourner une solution (i.e. une partition) quand le problème est satisfiable.

**Correction Question 1.**

1. l'algorithme retourne vrai si et seulement si $0 \leqslant p < m$ et s'il existe un sous-ensemble d'éléments du tableau $v$ dont la somme est égale à $p$ (chaque élément pouvant être sélectionné plusieurs fois). Sa complexité est $mn$ où $n$ est la taille du tableau $v$.
2. Le problème de partition consiste à décider s'il existe un sous-ensemble de S de poids total $\frac{\sum_{i \in S} w_i}{2}$. Si cette valeur est non entière alors le problème n'est pas satisfiable. Autrement : on note $p$ cette valeur, on numérote les objets de S de 0 à $n-1$ et on crée le tableau d'entiers $v[i] = w_i$ pour tout objet $i$, enfin on prend tout entier $m$ supérieur à $p$, $m = p+1$ par exemple. En inversant la boucle interne de `count`, le poids d'un élément n'est plus compté qu'une seule fois :

```
for (int k = m-1-v[i]; k >= 0; k--) {
   if (c[k]) {
     c[k+v[i]] = true;
   }
}
```

   Alors le problème est satisfiable si et seulement si `count`$(v, m, p)$ retourne vrai.
3. on a donc un algorithme pseudo-polynomial (dépend de $\sum_{i \in S} w_i$) pour résoudre ce problème. Le problème est donc soit dans la classe $\mathcal{P}$, soit dans la classe $\mathcal{NP}$-complet au sens faible. (Par réduction du problème subset-sum, on voit que partition appartient à cette seconde classe).
4. En plus du tableau de booléens c on maintient un tableau d'entiers `element` de taille $m$ initialisé à -1. La condition de la boucle est alors remplacée par :

```
if (c[k] && !c[k+v[i]]) {
   c[k+v[i]] = true;
   element[k+v[i]] = i;
}
```

   Si le problème est satisfiable, une partition de S est obtenue par l'algorithme suivant :

```
void solution(int[] element, int[] v, int p) {
   int[] partition = new int[v.length];
   while (p > 0) {
     int i = element[p];
     partition[i] = 1;
     p = p - v[i];
   }
}
```

   Une partition consiste à prendre tous les éléments $i$ de S tels que `partition[i]==1` d'une part et tous ceux tels que `partition[i]==0` d'autre part.

## 2 Modélisation linéaire et Heuristiques

**Problème 2** Gestion du traffic aérien.
*Le système de contrôle du traffic aérien d'un aéroport est chargé de répartir et de planifier l'atterrissage des avions sur les différentes pistes de l'aéroport de manière à minimiser les délais, tout en satisfaisant les contraintes de sécurité. Il s'agit de déterminer, pour chaque avion, sa piste d'atterrissage et sa date d'atterrissage, étant données les informations suivantes :*

- *il y a* $n$ *avions (notés* $i = 1, \dots, n$*) et* $p$ *pistes (notées* $k = 1, \dots, p$*) ;*
- *un avion* $i$ *ne peut atterrir sur une piste* $k$ *qu'à partir de la date* $e_{ik} \geqslant 0$ *;*
- *on note* $e_i$ *la date d'atterissage au plus tôt de l'avion* $i$*, c'est-à-dire la plus petite des dates* $e_{ik}$ *parmi toutes les pistes* $k$ *; le* ***délai*** *d'un avion* $i$ *est la différence entre la date d'atterrissage effective de l'avion* $i$ *et sa date d'atterrissage au plus tôt* $e_i$ *;*
- *si deux avions* $i$ *et* $j$ *atterrissent sur la même piste* $k$*, disons l'avion* $i$ *avant l'avion* $j$*, alors il doit se dérouler une durée minimale* $d_{ij} \geqslant 0$ *entre la date d'atterrissage de* $i$ *et la date d'atterrissage de* $j$ *;*

**Question 3** modèle compact ***(5 points)***
*On souhaite formuler ce problème par un premier modèle linéaire en nombres entiers compact* $(C)$ *basé sur les variables suivantes :* $t_i$ *la date d'atterrissage de l'avion* $i$*,* $x_{ik} = 1$ *si et seulement si l'avion* $i$ *atterrit sur la piste* $k$*,* $y_{ij} = 1$ *si et seulement si les avions* $i$ *et* $j$ *atterrissent sur la même piste et si l'avion* $i$ *atterrit avant l'avion* $j$*.*

**Q3.1.** écrire les formules logiques qui relient ces différentes variables entre elles, et les linéariser ;

**Q3.2.** écrire le modèle complet $(C)$.

**Question 4** modèle décomposé ***(5 points)***
*On souhaite formuler ce problème par un second modèle* $(P)$ *de set-partitionning. Pour cela, on suppose connue la valeur optimale* $c(s, k)$ *du problème restreint à un sous-ensemble d'avions* $s \subseteq \{1, \dots, n\}$ *et à une seule piste* $k \in \{1, \dots, p\}$*. Par définition,* $c(s, k)$ *est le délai minimal d'ordonnancement des avions de* $s$ *sur la piste* $k$*, indépendamment des autres avions sur les autres pistes.*

**Q4.1.** montrer que toute solution réalisable du problème général peut être représentée comme une partition de l'ensemble des avions en $p$ parties ;

**Q4.2.** exprimer alors le coût (délai total) d'une telle solution au moyen de la fonction $c$ définie ci-dessus ;

**Q4.3.** en déduire un second modèle linéaire en variables binaires $(P)$ du problème général.

**Question 5** heuristique de réparation ***(5 points)***
*On souhaite utiliser la relaxation continue de ce second modèle dans une méthode heuristique pour construire une bonne (non nécessairement optimale) solution au problème.*

**Q5.1.** étant donnée une solution fractionnaire optimale de la relaxation continue de $(P)$, décrire une méthode heuristique de construction d'une solution réalisable (entière) du problème initial à partir de cette solution fractionnaire ; ***Note*** *: si vous n'avez pas répondu à la question précédente : décrire une méthode heuristique constructive quelconque ;*

**Q5.2.** donner la complexité temporelle de pire cas de cet algorithme.

**Question 6** heuristique gloutonne : le cas 1 piste ***(5 points)***
*Plutôt que de calculer* $c(s, k)$ *pour tout sous-ensemble d'avions* $s$ *et toute piste* $k$*, on cherche à en calculer une évaluation par excès (borne supérieure) :*

**Q6.1.** décrire un algorithme glouton pour calculer une solution au problème avec une seule piste ;

**Q6.2.** en le couplant à l'heuristique de la question précédente, en déduire une méthode heuristique polynomiale pour le problème de gestion du traffic aérien.

**Correction Question 2.**

1.

$$
\begin{aligned}
(C):\ \min\ & \sum_{i=1}^{n}(t_i - e_i) \\
\text{s.t. } & t_i \geqslant e_i + (e_{ik} - e_i)x_{ik} && \forall i = 1,..,n, \forall k = 1,..,p, \\
& \sum_{k=1}^{p} x_{ik} = 1 && \forall i = 1,..,n, \\
& y_{ij} + y_{ji} \geqslant x_{ik} + x_{jk} - 1 && \forall i,j = 1,..,n, i < j, \forall k = 1,..,p, \\
& y_{ij} + y_{ji} \leqslant 1 && \forall i,j = 1,..,n, i < j, \\
& t_j - t_i \geqslant T(y_{ij} - 1) + d_{ij} && \forall i,j = 1,..,n, i \neq j, \\
& t_i \geqslant 0 && \forall i = 1,..,n, \\
& x_{ik} \in \{0,1\} && \forall i = 1,..,n, \forall k = 1,..,p, \\
& y_{ij} \in \{0,1\} && \forall i,j = 1,..,n, i \neq j
\end{aligned}
$$

2. Pour tout sous-ensemble d'avions $s \subseteq S = \{1,\dots,n\}$ et pour tout avion $i \in S$, on pose $\delta_i^s = 1$ si $i \in s$ et $\delta_i^s = 0$ sinon. On définit alors des variables binaires $x_{ks} = 1$ ssi les avions atterrissant sur une piste $k = 1,..,p$ sont les avions de l'ensemble $s \subseteq S$.

$$
\begin{aligned}
(P):\ \min\ & \sum_{k=1}^{p}\sum_{s\subseteq S} c(s,k)x_{ks} && && (1) \\
\text{s.t. } & \sum_{k=1}^{p}\sum_{s\subseteq S} \delta_i^s x_{ks} = 1 && \forall i = 1,..,n, && (2) \\
& \sum_{s\subseteq S} x_{ks} = 1 && \forall k = 1,..,p, && (3) \\
& x_{ks} \in \{0,1\} && \forall s \subseteq S, \forall k = 1,..,p. && (4)
\end{aligned}
$$

3. Soit $\bar{x}$ une solution fractionnaire. Step 1 : Pour chaque piste $k$, on choisit l'ensemble $s \subseteq S$ qui maximise $\bar{x}_{ks}$ (l'une au moins de ces variables est non nulle d'après (3)). On note $s_k$ l'ensemble sélectionné. Step 2 : Si un avion $i \in S$ appartient à plusieurs ensembles $s_k$, on choisit l'ensemble $k$ qui minimise $c(s_k,k) - c(s_k \setminus \{i\},k)$ et on supprime $i$ de tous les autres ensemble $s_{k'}$. Step 3 : Si un avion $i \in S$ n'appartient à aucun ensemble $s_k$, on choisit l'ensemble $s_k$ qui minimise $c(s_k \cup \{i\},k) - c(s_k,k)$ et on lui ajoute $i$. Au final, tout avion apparaît exactement une fois dans les ensembles $s_k$ : la solution est réalisable et de coût $\sum_{k=1}^{p} c(s_k,k)$. Les étapes 2 et 3 peuvent être fait en même temps dans une double boucle, sur les avions puis sur les pistes : $O(np)$. L'étape 1 nécessite de parcourir l'ensemble des variable $\bar{x}_{ks}$ non nulles, soit dans le pire des cas $O(|S|p) \leqslant O(2^n p)$.
4. Une approximation $c'(s,k)$ de $c(s,k)$ peut être obtenue par l'algorithme glouton suivant qui fait atterrir au plus tôt les avions de $s$ un par un : On choisit un avion $i_0 \in s$ qui minimise $e_{ik}$ (en cas d'égalité, on choisit celui qui maximise $\max\{d_{ji} \mid j \in s\}$), on retire $i_0$ de $s$ et on pose $t_{i_0} = e_{i_0 k}$. À l'étape suivante, on choisit un avion $i_1 \in s$ qui minimise $\max(e_{ik}, t_{i_0} + d_{i_0,i})$ (en cas d'égalité, on choisit celui qui maximise $\max\{d_{ji} \mid j \in s\}$), on retire $i_1$ de $s$ et on pose $t_{i_1} = \max(e_{i_1 k}, t_{i_0} + d_{i_0,i})$. À l'étape suivante, on choisit $i_2 \in s$ qui minimise $\max(e_{ik}, t_{i_0} + d_{i_0,i}, t_{i_1} + d_{i_1,i})$, etc.
5. Une heuristique constructive du problème complet pourrait consister à calculer un sous-ensemble de $S$, par exemple : un ensemble des $s \in S$ de cardinalité $|s| = n/p$ (on peut supposer que chaque piste recevra en moyenne le même nombre d'avions), et de calculer la borne supérieure $c'(s,k)$ pour chacun de ces $s$ et pour chaque piste $k$. On résoud alors la relaxation continue du second modèle sur ce seul ensemble de colonnes et avec un coût approché. On applique enfin l'heuristique de réparation sur la solution fractionnaire obtenue.